SCRIPT PROGRAMMING G1F

Spring Term 2016

Assignment 2 – PowerShell

Alexander Kratzsch ⟨alexander.kratzsch@his.se⟩
Rikard Sandelin
Magnus Axelsson ⟨magnus.axelsson@his.se⟩

Aim
The lab assignment aims to provide the student skills in:

- Use regular expressions to match patterns
- Systematically design, implement, and troubleshoot scripts based on problems related to system administration

Requirements
To achieve a G on this assignment:

- Complete all G-level parts of the assignments
- Demonstrate all your solutions
- Submit your source code, design report, and bug report in SCIO
- Format and comment the source code according to the instructions

To achieve a VG on this assignment:

- Fulfill the requirements for a G as described above
- Complete all VG-level parts of the assignments
- Demonstrate all your solutions
- Submit your source code in SCIO
- Format and comment the source code according to the instructions

Contents

1 INTRODUCTION

This document contains instructions for the lab assignment on PowerShell in the course 'Script Programing'. After the introduction with details about examination and submissions as well as literature the lab instructions has a 'Get comfortable' part that is designed to help you start using PowerShell. The assignment then has three G-level parts and one VG-level part.

1.1 Examination

The PowerShell lab assignment covered here examines Assignment 2 (*Inlämningsuppgift 2*) of the course. The assignment is graded U/G/VG. To score a G on this assignment you must complete all G level parts of the assignment, to score a VG you also need to pass the VG part. All scripts must be demonstrated to a lab supervisor. The source code for all scripts and associated documents must be submitted through SCIO before the deadline as a single .zip file; omit any backup files.
General requirements for examination of this assignment:

- Your scripts has to be fully functional in compliance to the provided guidelines.
- Your scripts must be compliant with the requirements stated for each script.
- Your source code must be properly formatted (*indenterad*).
- Your source code must contain relevant comments that explain your code. Please note that correct spelling and grammar is required.
- Your variables and functions must have (self-)descriptive names.
- If your script produces any output it should be relevant and formatted.
- Your script may not clutter up the output with error messages. You should employ error-handling.
- All scripts must contain the following metadata-field:

  ```
  ################## METADATA ##################
  # NAME: your full name
  # USERNAME: your login name
  # COURSE: this course's name
  # ASSIGNMENT: name and number of assignment
  # DATE OF LAST CHANGE: date in ISO~8601
  ##############################################
  ```

- All scripts must contain `Set-StrictMode`. Scripts must run without error or warning messages. Errors in the program should be handled gracefully and presented to the user.
- All scripts must be able to run on the lab environment used in this course.
- Modules may not ask the user for any input or produce output. Input and output should happen in the scripts that make use the modules to perform specific tasks. This communication between script and module should be realized by using arguments and return values without global variables.

1.2 Lab Environment

You will download and install a OVA-file, found here:
`http://files.nsa.his.se/labfiles/it341g/ad-server.ova`. This file contains a Windows Server 2012 with an installed and populated AD and other services. Use the IP address `10.⟨room⟩.⟨desk⟩.11`. You will also install a virtual Windows 8.1 (**not** the *N* variant) machine that should be joined to the domain. To make operation more smooth you may increase the resources for the machine. You should set your `ExecutionPolicy` to `RemotelySigned` on all clients. You are recommended to use Windows PowerShell ISE (Integrated Scripting Environment) or another editor that supports PowerShell scripting (such as Notepad++).

1.3 Connection to Course Plan

This assignment, fully or partially, examines the following course goal:

- Systematically design, implement, and troubleshoot scripts based on problems related to system administration

1.4 Cheating

All assignments must be completed individually and all students must submit their own created work. You are allowed and encouraged to discuss solutions with each other, but it is strictly forbidden to copy each other's work. Copying each other's work is considered cheating.
If we suspect cheating, we must and will report that to the head of division. Suspicion of cheating will result in a meeting with the head of division and, once considered to be a valid case, will lead to a hearing at the disciplinary board. Contact your supervisor or course coordinator if you have any questions.
Before you submit your tasks you must read Appendix A. By submitting or demonstrating (parts of) your work you acknowledge that you have read and understood Appendix A.

1.5 Contacts

This assignment will be supervised by:

- Alexander Kratzsch ⟨alexander.kratzsch@his.se⟩
- Rikard Sandelin
- Magnus Axelsson ⟨magnus.axelsson@his.se⟩

Good luck with the lab!

2 DOCUMENTATION

For this assignment there is plenty of great documentation available. Here are some pointers:

- The official PowerShell documentation:
  `http://technet.microsoft.com/en-us/library/bb978526.aspx`
- Swedish book: Kom igång med PowerShell
- Free books about PowerShell: `http://it-ebooks.info/tag/powershell/`

There are two cmdlets that is of great help when trying to find a way of doing a task in Powershell:

- `Get-Help` provides a help page for commands (such as cmdlets) containing a description of the command, a list of available parameters and examples of usage (see the `-examples` parameter).
- `Get-Command` can be used to search for commands and to get information of their type (cmdlet, alias, etc). You may for example use the `-verb` and `-noun` parameters to search for commands containing a certain word.

3 REPORT

You have to submit a description together with your scripts that explains the design of your scripts. This text should give a general explanation of your scripts function as well as design decisions you have made during development. Examples of design decisions are if/which data structures you have used to store information or when/if/why you used/didn't use functions. You are expected to shortly reflect about whether the execution of scripts should be allowed on client computers, from a system administrators and a users point of view. Discuss also if signing of scripts provides enough security. Are there other aspects that should be considered as well?

4 ADMINISTRATION OF USERS

4.1 Create Automatically User Names and Passwords

Your Windows Server 2012 Active Directory Service is already populated with a number of users. For this assignment you will need to add even more users. A common scenario is that you as a system administrator get a Excel document from the Human Resources department with the new employees. These users are then to be imported into the AD. The fastest way is to convert the document into a CSV file and import it with a script. For this assignment, you can download a CSV file from `http://www.nsa.his.se/labfiles/IT341G/users.csv`.
Your task is to write a script that:

- Automatically creates usernames (`sAMAccountName`)

  Please note that the `sAMAccountName` needs to be unique within the domain.

- Automatically generates other attributes in a way that follows the structure already in use in the Active Directory domain:
  - `name` should be the combination of firstname and lastname
    Please note that the `name` attribute needs to be unique within the organizational unit.
  - `userPrincipalName` should be the *`sAMAccountName`*`@scripting.nsa.his.se`
    Please note that the `userPrincipalName` needs to be unique within the forest.

- Automatically create 'random' password

- Fill in the email address (the same as `userPrincipalName`)

- Enable the account

- Force users to reset their password on first login

- Create a small text file for each user, named after the user and containing their login credentials (username and password)

- Add users to appropriate groups (and if needed create additional groups)
  - Shadow groups
  - Localisation groups

- If no CSV-file is given on the command line when the script is invoked it should prompt the user for a CSV-file.

4.2 Automatically Update Shadow Groups

Shadow groups are groups that contain users in a certain organizational unit in the AD. There is no automatic way to synchronize the users in a OU with the corresponding shadow group. Your next script should fix this problem. It should automatically check if any of the shadow groups need to add or remove members, and conduct the necessary changes. In other words it should make sure that the shadow groups and the corresponding organizational units are in sync. It is the OU that decides in which shadow group a user should be placed. The script should be scheduled to run automatically every hour, on the full hour. The scheduling of the script does not have to occur via a script though.

5 ADMINISTRATION OF BACKUPS

5.1 Multiple Backups to Network Share

There should be a second partition on your AD-server `DC01` that is mounted on the drive letter `B:` and called `Backup`. It is shared and should be accessible on the network `\\DC01\Backup`. You will create a script that enables you to save multiple backups to a network share. Your script should utilize Windows Server backup to perform the actual backups, but the script should enable you to change where the backup is taken depending on the current date. This is necessary since a backup to a network share usually replaces earlier backups in the same folder.
Your script is supposed to:

- Be scheduled to run every night at 01:30
- Relevant output has to be written to the EventLog
- Prepare the Backup network share for the backups
  - Create new year-folders as necessary
  - Create new month-folders as necessary
  - Create new weekday-folders as necessary
  - If it is Sunday and there already exists a backup in the weekday-folder, the old backup needs to be moved to `\\DC01\Backup\$YEAR\$MONTH\$HOSTNAME-$YEAR-$MONTH-$DAY`
- Create a Windows Server Backup that
  - Takes a backup of the SYSVOL-folder of your `DC01`
  - This backup should be placed in the Network share `\\DC01\Backup\$YEAR\$MONTH\$WEEKDAY`

6 REMOTELY COLLECT INFORMATION FROM ALL DOMAIN COMPUTERS

6.1 Inventory Script

Collecting information from all clients in a domain is hard manual work. You will therefore create a script that will automatically collect the information for you. To get more than one computer to inventory, please join the domain with your physical client computer.
Your script is supposed to:

- Get operating system and version for all computers (desktops) in the domain.
- Get last date updates were installed for all computers (desktops) in the domain.

- Get total disk space for all computers (desktops) in the domain.
- Get free disk space for all computers (desktops) in the domain.
- Format this in a table and output to a file. The file name and location should be given by the user.

To be able to do this properly you will most likely have to enable Remote Administration (RPC) through your clients' firewalls – you should set this up via a GPO (Remote Service Management).

Note: Have a look at `http://technet.microsoft.com/enus/library/dd819505.aspx` and `http://technet.microsoft.com/en-us/library/dd315240.aspx`.

6.2 Create a Script for Users

In an enterprise environment you might want to be able to run scripts on your clients, but you do not want all scripts, especially ones downloaded from the Internet, to be executed. You will therefore enforce a `ExecutionPolicy` of `Allsigned` on all of your clients. This should be done with the help of a GPO.
You will now create a script that will help users diagnose their disk space status.
Your script is supposed to:

- Report to the user how much disk space he is using. Basically the script should tell the user the size of the `C:\Users\$USER` directory.

Place your script in `\\DC01\Shared`.
You also need to sign your script and then make sure that the signature is recognized as a trusted one. You will use the installed Active Domain Certificate Service to produce a certificate to use for signing. Make use of a GPO to place your signature as trusted on your client computers. Please refer to the following guide for this task:

1. `http://blogs.technet.com/b/heyscriptingguy/archive/2010/06/16/hey-scripting-guy-how-can-i-sign-windows-powershell-scripts-with-an-enterprise-windows-pki-part-1-of-2.aspx`
2. `http://blogs.technet.com/b/heyscriptingguy/archive/2010/06/17/hey-scripting-guy-how-can-i-sign-windows-powershell-scripts-with-an-enterprise-windows-pki-part-2-of-2.aspx`

7 CREATE YOUR OWN CMDLET (VG)

To get a VG in the PowerShell part you have to build your own cmdlet. This cmdlet should be able to create a new shared directory for you. The cmdlet should be controlled by parameters, both mandatory and optional ones.

Note: Have a look at the list of standard cmdlet parameter names: `http://msdn.microsoft.com/en-us/library/dd878352(v=vs.85).aspx`. If a fitting parameter name exists on that list please use it.

The created shared directory should:

- Have a name and path.
- Be shared on the network.
  - Under what name?
  - Valid share permissions.
- Have valid ACLs

  - Who should have access?
- Error-handling?
  - What happens if the folder already exists?
- You are welcome to implement your own ideas

You might want to check out this blog-series: `http://blogs.technet.com/b/heyscriptingguy/archive/tags/windows+powershell/guest+blogger/sean+kearney/build+your+own+cmdlet/`
Submit your reasoning for your choice of parameters together with your source code. Your submission for Part VG must also contain a brief text that explains how you designed your cmdlet (i. e. how you stored your variables, when/if/why you used/didn't use functions etc.) and what bugs you encountered and how you solved those (i. e. had issues with encoding of files used as input for the script, solved it by…).

A PLAGIARISM AND CHEATING

This appendix contains information about rules regarding cheating/plagiarism and some hands-on advice for academic writing. As long as nothing else is explicitly stated in written form – these rules apply to all assignments in this course.

A common rule for all assignments during academic studies is that you may only submit your own, individual work. Submitting assignments that are fully or partially copied from or created by someone else is strictly forbidden.

If a teacher who is grading assignments suspects cheating/plagiarism, he/she is required to report this to the head of division. If the head of division confirms the teacher's suspicion, the case will be forwarded to the University's disciplinary board, which will investigate the case and summon both the student and the teacher in an official hearing. Based on its findings, the board may close the investigation, reprimand the student, or suspended the student for up to six months.

A video presentation about cheating/plagiarism has produced by Marcus Nohlberg who is a teacher at this university:
`https://www.youtube.com/watch?v=PPyfviclQF0` (in Swedish)

A.1 References

As a student you are encouraged to back up your work with the work of others using references. There is a big difference between plagiarism and using references. Plagiarism implies that you copy the work of someone else without giving him/her credit for it. It also implies that you copy larger portions of text. To make sure that you are never suspected for plagiarism/cheating always write your text in your own words and use references whenever appropriate. You are allowed to include shorter citations in your work, when using citations you must include a reference.

If you want to learn more about academic writing, reference management etc., the library provides you with access to a lot of resources. You may also sign up for a course in academic writing.

A.2 Co-operation

You are allowed to help each other out. However there is a difference between helping someone and doing someone's work. Even if you are encouraged to discuss solutions to the assignments you must write your own code and understand what your code does.

It is usually a bad idea to give a copy of your (complete) report to another student in order to 'help' him/her. It is happening far too often that the receiver of your report will copy&paste parts of your report and suddenly you are involved in a case of plagiarism (see above). You may even receive a warning for handing out your report all too quickly.